Intelligent Agile Method Framework

Marko Janković, Marko Bajec, Ghazaleh Khodabandelou, Rebecca Deneckere, Charlotte Hug, Camille Salinesi

Intelligent Agile Method Framework

Marko Janković[1], Marko Bajec[1], Ghazaleh Khodabandelou[2], Rebecca Deneckere[2], Charlotte Hug[2] and Camille Salinesi[2]

[1]*Faculty of Computer and Information Science, University of Ljubljana, Tržaška cesta 25, SI-1000 Ljubljana, Slovenia*

[2]*Centre de Recherche en Informatique, University of Paris 1 Panthéon-Sorbonne, Paris, France*

{*marko.jankovic, marko.bajec*}*@fri.uni-lj.si,*

{*ghazaleh.khodabandelou, charlotte.hug, rebecca.deneckere, camille.salinesi*}*@univ-paris1.fr*

Keywords: Situational Method Engineering; Software Development Improvement; Process Mining

Abstract: The paper addresses the problem of the low usage of software development methods in software development practice. This has been recognized as one of the key reasons for failures in software development projects and a contributor to the low quality of software. We introduce a novel approach that could help to improve the maturity of software development processes. The approach is based on the method engineering principles taking into account the limitations that hinder its use in practice. The main objective of our research is to show that the method engineering concepts are applicable in real settings and that could contribute to the higher quality of software development processes and their products.

1 INTRODUCTION

According to the literature, the use of software development methods (SDM) in software industry proves beneficial as it contributes to the higher quality of the process and its product, i.e. the developed software (van de Weerd et al., 2006), (Bekkers et al., 2008), (Bajec et al., 2007), (Karlsson and Ågerfalk, 2004), (Fitzgerald and Hartnett, 2005). Despite these evident benefits, the studies on the maturity of the software development discipline show that a large percent of software development companies do not have their SDMs documented and those that have, do not really follow them or do not follow them rigorously when developing software. Several reasons have been identified in the past to explain this behavior. Two of them seem to be most important: (1) rigidity, which does not allow SDMs to be adapted to specifics of a particular project, and (2) their social inappropriateness – i.e. prescribed methods are not based on the knowledge, culture and attitude of the developers and other stakeholders of the project (Riemenschneider et al., 2002), (Mohan and Ahlemann, 2011), (Mirbel and Ralyté, 2006), (Karlsson and Ågerfalk, 2004).

The above phenomenon has been widely acknowledged in the research community and to some extent also among the practitioners. The related problems are severe. According to the research conducted by McKinsey in collaboration with the University of Oxford (Bloch et al., 2012), half of all large IT projects (projects with initial price exceeding $15 million) run over budget, over time or deliver less value than predicted. Furthermore, study conducted by Standish Group (www.standishgroup.com), compiled into the CHAOS Report 2009, reveal that only 16% of observed projects have been finished in time; 32% were terminated before they were completed; and 52% were completed behind their schedule and involved higher cost. All these facts show that this is not an insignificant problem and that there is a need for new ideas and approaches to confront with these challenges.

In this paper, we describe an approach that could help to improve the maturity of software development processes by circumventing the problems that hinder the use of SDMs in practice. On the long run our rather ambitious goal is to lower the risk for IT project failures and to improve the overall quality of the developed software.

The approach is based on the method engineering principles and represents a continuation of our past research in this field (Bajec et al., 2007), (Vavpotic and Bajec, 2009), (Žvanut and Bajec, 2010), (Hug et al., 2012), (Deneckere et al., 2008).

In the paper we outline the main idea (Section 3), provide its background (Section 2), describe the research method (Section 4), and finally explain the remaining steps to be done (Section 5).

2 BACKGROUND

For decades, researchers have been striving to develop new approaches and solutions as a response to low usage of SDMs in practice. We can see the emergence of configurable methods and supporting tools (e.g. IBM Rational Method Composer (RMC), Microsoft Solutions Framework) that support configuration and tailoring of SDMs to better suite the characteristics of a particular project and organization (Garcia et al., 2011). However, although these tools often also include best practices, formed on the basis of vast experiences in software development, they still require specific knowledge in order to be used efficiently. To configure a method using IBM Rational Method Composer, for example, we need to know RUP in great details.

Another related domain is light and agile methods, which are more user-oriented, less complex and thus easier to implement and use (Laanti et al., 2011), (Dybå and Dingsøyr, 2008). In general, agile and light methods are more adoptable, as they strive to be closer to developers and do not bring such overhead as traditional methods. But the fact is that companies keen of agile and light methods rarely use specific methods but rather just follow their vision. This again leads to the fact that their ways of working are not documented and that valuable knowledge gained on development projects remain only in the heads of individual developers.

Very related to our work is the discipline called Method engineering (ME). Brinkkemper (1996) has defined ME as an engineering discipline to design, construct and adapt methods, techniques and tools for the development of information system. A specific sub-discipline of ME is situational method engineering (SME), which tends to support creation of methods from method parts or tailoring existing methods to meet requirements of a specific project. A number of approaches of SME (Ralyté and Rolland, 2001), (Brinkkemper et al., 1998), (Deneckere et al., 2008), have been proposed and developed, but despite the fact that these methods are designed and elaborated to be adapted, empirical researches show that their application in practice is rare (Mirbel and Rivieres, 2002), (Ralyté and Rolland, 2001). A reason for the latter can be found in the fact that creating and tailoring of methods is time-consuming and requires a considerable commitment of the developers and other stakeholders. Furthermore, to apply SME we need a person that is familiar with numerous methods and thus capable of constructing new methods by composing different fragments together. Furthermore, SME approaches often do not consider social and technical aspects. This leads to prescribed methods that are socially or technically noncompliant with the organization and developers, and consequently not accepted. As an answer to that issue Bajec et al. (2008) have developed an innovative approach, called Agile Method Framework (AMF). Main vision of this approach was to bring SME principles closer to practitioners, by considering social and technical aspects that influence on how practitioners perceive usefulness of methods as guidelines for their everyday work. Although the results showed that AMF was in practice perceived as useful, it was not fully accepted in the participating software companies, due to the fact that it still required the active involvement of developers. A more detail review of the past research in the field of SME can be found in the work of Henderson-Sellers et al. (2010) .

3 INTELLIGENT AGILE METHOD FRAMEWORK

The main idea of the approach that we call Intelligent Agile Method Framework (in further text iAMF) is to provide methods and tools to (a) help companies capture the software processes they are de-facto using (i.e. the real ways of working when developing software), (b) help them follow these processes consistently and finally (c) help them to continuously improve and document what they do on IT projects. While there are other approaches striving to the same goal, the peculiarity of our approach is that it will require only insignificant effort from the developers and other stakeholders of the project. The latter is one of the main reasons why similar initiatives and approaches have not succeeded to penetrate in real practice (Mirbel and Ralyté, 2006), (Karlsson and Ågerfalk, 2012).

The first step in iAMF is to capture how projects are typically performed in a particular company. Our hypothesis is that if prescribed methods are not just some off-the-shelf methods (such as RUP, SSADM etc.) but rather based on the knowledge and experiences of the company employees, and take into account the real ways of working in the company, then it is much more likely that developers will follow them rigorously. Instead of involving developers to describe how they do projects, iAMF will learn from the observation on what developers and other stakeholders actually do on the projects, how they perform and act, and based on that create (document) base methods. With a base method we denote a semi-formal description of how a certain company is performing its projects. Note that a particular company

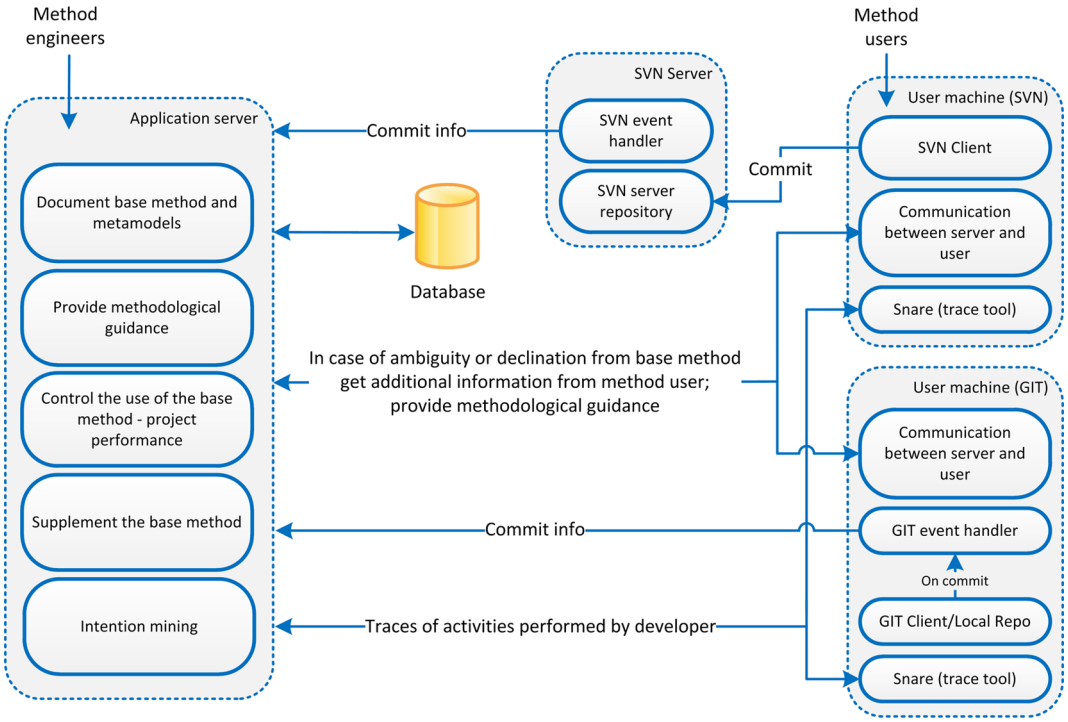

Figure 1: High-level architecture of iAMF supporting toolset

might employ different software processes for different projects. In this sense, the base method represents an overall method that comprises all the different processes that a company is using when performing its projects. With the first step of the iAMF the respected software process maturity rises to level 3 (documented).

Once base methods are captured, iAMF can be used to supervise project performance and guide developers reusing the knowledge and past experience. Any declination from base methods would be detected and either prevented or treated as a new knowledge and used to supplement the base method. By this, iAMF facilitates continuous process improvement, which helps to raise the maturity of the process to level 4 (managed) or even 5 (optimized).

3.1 iAMF supporting toolset

In this section, a high-level architecture of the iAMF toolset is presented (Figure 1). The purpose of the toolset is to facilitate the iAMF and will be used in the evaluation phase (see the section on Research Approach) to validate our approach. Further, we briefly describe the toolset main components, their purposes and communication.

3.1.1 Documenting base methods and metamodels

To support the enactment of software process following some prescribed SDM, the first step is to capture and document the prescribed method, i.e. the base method. Even though it is generally accepted that elaboration of SDM have a positive influence on the software development process (e.g. for the sake of knowledge reuse and quality assurance), companies very much differ in the level of details they believe is useful to capture and document (García et al., 2011). Furthermore, keeping ways of working documented and up-to-date is “expensive” and time-consuming. One of the goals of the iAMF toolset is to enable efficient and user friendly capturing of base methods and metamodels that underpin them. Companies can select from metamodels that support known methods, such as RUP, and adapt them to suit their purposes and specifics, or to create new metamodels from scratch (Bajec and Vavpotic, 2008). In the later phases metamodels are crucial to check completeness and consistency of the documented base methods. The toolset also supports the creation of the map intentional process models for intention mining purpose (see Intention mining). The described functionality is available through graphical user interface, which allows method engineers to construct or update base methods

and metamodels, simply by dragging and dropping elements on the screen and connecting them to express their relationships. For each metaelement, method engineers can enter detailed information, including templates that are then used to guide the documentation of their instances.

3.1.2 Intention mining

Intention Mining aims to infer user intentions from event logs (Khodabandelou et al., 2013b). Intention mining uses event logs as input and allows (1) identifying intentional process models, (2) checking the conformity between a prescribed intentional model and its enactment, (3) identifying the gaps between the model and the traces and (4) providing recommendations to users at run-time, based on their reasoning behind their activities (Khodabandelou et al., 2013a).

Intention mining is an important part of the iAMF toolset. Its purpose is to gather information on intentions the developers were following when performing various tasks and activities on past projects. For this purpose trace-based tools will be employed. Instead of developing a completely new tool, we conducted a research (Khodabandelou et al., 2013c), which showed that open source tool Snare could be efficiently used for our purposes, as it covered almost all our requirements. The Snare will be used to collect traces of activities and retrieve information about what and how developers perform during projects. Our expectation is that by the analysis of traces we will be able to reconstruct intentions that guided developers when performing specific tasks and based on that create intentional process models for further guidance.

3.1.3 Providing methodological guidance

Once a base method and metamodels are captured, the guidance can be provided to developers who are novice or not familiar enough with the base method their company is prescribing. To improve and personalize methodological guidance, collected traces will be used to automatically categorize users according to their profiles. Based on that and together with the context at hand and history records, the users will be provided with an advice on what step to take next in a particular situation. Furthermore, process models combined with examples of best practices will guide developers to efficiently perform their activities. In order to give users good overview and efficient guidance, the iAMF toolset will provide graphical interface for users to browse between various base methods, to get detailed information on a particular element, or just to scroll through the base method visualized as a process diagram or a navigation tree.

3.1.4 Controlling the use of base method

The most important feature of iAMF is the ability to observe the method in action and to identify any deviation from the method the team members are expected to follow (base method). To detect deviations, the proposed framework relies on two sources of information: (1) the base method that prescribes how different kinds of projects should be carried out (see also the section Documenting Base Methods) and (2) the real method or method in action, which is not documented but can be inferred by observing what developers do and how they perform on projects.

To infer the in-action method we will intercept every action a user does on the Revision control system (RCS) (i.e. checking an artifact in or out), and on the basis of (1) information gained from collected traces, which hold information about the activities performed before the action, (2) information extracted from the content of the artifact with text mining, and (3) information mined from RCS logs, predict current activity and link it with the base method. If the action will not be in accordance with the base method, the developer performing the action will be asked to provide the explanation for the deviation. Sometimes the deviation will be unintentional and the developer will be instructed to follow the prescribed method while in other cases the deviation will be a result of a specific situation not seen before and will be thus used to supplement the base method. For the beginning, only support for Subversion and GIT that are major revision control systems in use today, will be implemented. However, a generic interface will be provided to support the information capture from a variety of application and systems developers are using to facilitate their work in software development.

With all the collected data we expect to be able to infer the "path" that has been taken through the base method as well the current stage of the project according to the base method (e.g. which activities and artifacts have already been performed/produced and what steps remains to be done). The collected data will also be used to provide guidance for the developers on the project.

3.1.5 Supplementing the base method

The deviations from base methods might have different reasons. It might happen, for example, that the user made a mistake, e.g. by taking some action that was not expected according to the base method – i.e.

he checked in an artifact A, which was not yet required, while some other artifact B, which would need to be created before the A, was not yet in the RCS. But on the other hand, there might be specific circumstances that leaded the user to react differently than prescribed by the base method. In the first case, the user is expected to eliminate or compensate the mistake, e.g. by performing the activity he forgot and by creating the missing artifact, while in the latter he has to supplement the base method so that the new circumstances including possible new method elements and relations would be covered for the future cases. In this way the base method continuously complements and improves.

4 RESEARCH APPROACH

The research positioned in this paper presents a join work of researchers from the University of Ljubljana and University of Paris 1. Both research groups are experienced in the SME research field with good research record. Their joint concern is to improve software development practice by the application of the SME approaches and principles. Having good connections and experience with local industry, the teams are seeking for cross-cultural collaboration that could provide deeper understanding of the subject matter and provide solutions that could be useful beyond their local environments.

In this section we briefly outline the main phases of the research:

Phase 1: Initiation. In the initiation phase we will take care for the research environment. We will select the RCS systems that are most popular among software companies so that we could then develop interfaces to capture required information. Similarly, a workflow-based enactment system will be selected to be able to get detailed information on software processes and specifically on developers' behavior when performing tasks within these processes. As a part of the initiation phase we will try to identify and stimulate companies for their participation on the project. To be able to evaluate the suggested approach in real settings, we will invite several software companies from Slovenia and France to participate in the research. To cover the variety of the software industry, we will try to involve companies of different sizes, organizations, cultures, experiences, ...

Phase 2: Implementation. In this phase we will implement the required iAMF tools. These will include all the components described in section 3, including algorithms for mining intentions from process logs.

Phase 3: Testing. The iAMF will be first tested within the university environment (at both universities). A group of students will use the framework as a part of their seminar work. The purpose of these tests will be to detect and eliminate software bugs, as well as to get general opinion on the usability of the framework. iAMF will be then improved based on the testing results.

Phase 4: Evaluation. In the evaluation, our goal is to empirically confirm the usability of the approach. This is not an easy task, as it cannot be performed within university environment. To get reliable feedback, several companies will be invited to evaluate the framework. With the evaluation we will try to confirm the main hypotheses: (H1) If prescribed methods were based on real ways of working and up to date then developers would follow them much more rigorously. (H2) iAMF can provide efficient support to project members by helping and guiding them in software development projects. (H3) Developers and other users do not perceive iAMF as an additional burden and are willing to use it. (H4) By using iAMF, companies can raise the maturity of their software processes to at least level 3.

5 CONCLUSION

In this position paper we outlined a research that aims at providing a novel approach for improving software development practice. The approach consists of a set of methods and tools that will help companies to elaborate and continuously improve their software processes, which will in-turn make them more useful and usable for developers and other stakeholders on software development projects. The research proposed in this paper is due to its time-consuming evaluation phase longitudinal. Giving any concrete conclusion at this moment would be premature and risky.

REFERENCES

Bajec, M. and Vavpotic, D. (2008). A framework and tool-support for reengineering software development methods. *Informatica, Lith. Acad. Sci.*, 19(3):321–344.

Bajec, M., Vavpotič, D., and Krisper, M. (2007). Practice-driven approach for creating project-specific software development methods. *Information and Software Technology*, 49(4):345–365.

Bekkers, W., van de Weerd, I., Brinkkemper, S., and Mahieu, A. (2008). The influence of situational factors in software product management: An empirical study. In *Second International Workshop on Software*

Product Management, 2008. IWSPM '08, pages 41 – 48.

Bloch, M., Blumberg, S., and Laartz, J. (2012). Delivering large-scale it projects on time, on budget, and on value. Retrieved from http://www.mckinsey.com/insights/business_technology/.

Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and Software Technology*, 38(4):275–280.

Brinkkemper, S., Saeki, M., and Harmsen, F. (1998). Assembly techniques for method engineering. In *Proceedings of the 10th International Conference on Advanced Information Systems Engineering*, CAiSE '98, page 381–400, London, UK, UK. Springer-Verlag.

Deneckere, R., Iacovelli, A., Kornyshova, E., and Souveyet, C. (2008). From method fragments to method services. In *Proc. Of EMMSAD'08*, pages 80–96, Montpellier, France.

Dybå, T. and Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9–10):833–859.

Fitzgerald, B. and Hartnett, G. (2005). A study of the use of agile methods within intel. In Baskerville, R., Mathiassen, L., Pries-Heje, J., and DeGross, J., editors, *Business Agility and Information Technology Diffusion*, volume 180 of *IFIP International Federation for Information Processing*, pages 187–202. Springer US.

Garcia, F., Vizcaino, A., and Ebert, C. (2011). Process management tools. *IEEE Software*, 28(2):15 –18.

García, J., Amescua, A., Sánchez, M.-I., and Bermón, L. (2011). Design guidelines for software processes knowledge repository development. *Information and Software Technology*, 53(8):834–850.

Henderson-Sellers, B. and Ralyté, J. (2010). Situational method engineering: State-of-the-art review. *Journal of Universal Computer Science*, 16(3):424–478.

Hug, C., Deneckère, R., and Salinesi, C. (2012). Map-tbs: Map process enactment traces and analysis. In *RCIS*, pages 1–6.

Karlsson, F. and Ågerfalk, P. J. (2004). Method configuration: adapting to situational characteristics while creating reusable assets. *Information and Software Technology*, 46(9):619–633.

Karlsson, F. and Ågerfalk, P. J. (2012). MC sandbox: Devising a tool for method-user-centered method configuration. *Information and Software Technology*, 54(5):501–516.

Khodabandelou, G., Hug, C., Deneckère, R., and Salinesi, C. (2013a). Process mining versus intention mining. In *Procs. EMMSAD'13*, Valencia, Spain.

Khodabandelou, G., Hug, C., Deneckère, R., and Salinesi, C. (2013b). Supervised intentional process models discovery using hidden markov models. In *Procs. RCIS'13*, Paris, France.

Khodabandelou, G., Hug, C., Deneckère, R., Salinesi, C., Bajec, M., Kornyshova, E., and Janković, M. (2013c). A systematic review and selection of cots products to trace method enactment. In *Proceedings of the 21st European Conference on information Systems*, Utrecht, Netherland.

Laanti, M., Salo, O., and Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at nokia: A survey of opinions on agile transformation. *Information and Software Technology*, 53(3):276–290.

Mirbel, I. and Ralyté, J. (2006). Situational method engineering: combining assembly-based and roadmap-driven approaches. *Requirements Engineering*, 11(1):58–78.

Mirbel, I. and Rivieres, V. d. (2002). Adapting analysis and design to software context: The JECKO approach. In Bellahsène, Z., Patel, D., and Rolland, C., editors, *Object-Oriented Information Systems*, number 2425 in Lecture Notes in Computer Science, pages 223–228. Springer Berlin Heidelberg.

Mohan, K. and Ahlemann, F. (2011). What methodology attributes are critical for potential users? understanding the effect of human needs. In Mouratidis, H. and Rolland, C., editors, *Advanced Information Systems Engineering*, volume 6741 of *Lecture Notes in Computer Science*, pages 314–328. Springer Berlin / Heidelberg.

Ralyté, J. and Rolland, C. (2001). An assembly process model for method engineering. In Dittrich, K. R., Geppert, A., and Norrie, M. C., editors, *Advanced Information Systems Engineering*, number 2068 in Lecture Notes in Computer Science, pages 267–283. Springer Berlin Heidelberg.

Riemenschneider, C., Hardgrave, B., and Davis, F. (2002). Explaining software developer acceptance of methodologies: a comparison of five theoretical models. *IEEE Transactions on Software Engineering*, 28(12):1135 – 1145.

van de Weerd, I., Brinkkemper, S., Souer, J., and Versendaal, J. (2006). A situational implementation method for web-based content management system-applications: method engineering and validation in practice. *Software Process: Improvement and Practice*, 11(5):521–538.

Vavpotic, D. and Bajec, M. (2009). An approach for concurrent evaluation of technical and social aspects of software development methodologies. *Information and Software Technology*, 51(2):528–545.

Žvanut, B. and Bajec, M. (2010). A tool for IT process construction. *Information and Software Technology*, 52(4):397–410.